Question 1 (5+5=10 points)

(A) Based on the description provided, identify an appropriate data structure:

- Items stored in this structure are located in contiguous locations in the computer’s memory.
  Answer: array or ArrayList

- The number of items stored in this contiguous storage, sequential structure can be determined by the length attribute.
  Answer: array

- The number of items stored in this contiguous storage, sequential structure can be determined by the predefined size() function.
  Answer: ArrayList

- Items added to this structure can only be removed in the last-in-first-out order.
  Answer: Stack

- Items added to this structure can only be removed in the first-in-first-out order.
  Answer: Queue
Question 1 (5+5=10 points)

- (B) Match (by drawing lines) the items on the left with complexity characterizations on the right as appropriate:

![Complexity Characterizations Diagram]

Question 2 (5+5+5=15 points)

A variable L which is an instance of a class that implements the List<E> interface is used to store items using the following:

```java
for (int i=1; i <= 5; i++) {
    L.add(2*i);
}
```

In the pictures below, clearly identify all essential components of the data structure used.

(A) Suppose L is defined as shown below:

```java
ArrayList<Integer> L = new ArrayList<Integer>();
```

Draw a picture of the resulting structure showing the items added above:
A variable L which is an instance of a class that implements the List<E> interface is used to store items using the following:

```java
for (int i=1; i <= 5; i++) {
    L.add(2*i);
}
```

In the pictures below, clearly identify all essential components of the data structure used.

(A) Suppose L is defined as shown below:

```java
ArrayList<Integer> L = new ArrayList<Integer>();
```

Draw a picture of the resulting structure showing the items added above:

(B) Suppose L is defined as shown below:

```java
LinkedList<Integer> L = new LinkedList<Integer>();
```

Draw a picture of the resulting structure showing the items added above:
**Question 2 (5+5+5=15 points)**

A variable L which is an instance of a class that implements the List<E> interface is used to store items using the following:

```java
for (int i=1; i <= 5; i++) {
    L.add(2*i);
}
```

In the pictures below, clearly identify all essential components of the data structure used.

(C) Suppose L is defined as shown below:

```java
Queue<Integer> L = new LinkedList<Integer>();
```

Draw a picture of the resulting structure (it is a linked list, with head and tail stored, front of queue is at tail and rear is at head) showing the items added above:

![Linked List Diagram](image.png)

**Question 3 (10+5 = 15 points)**

Given the class Place as defined in your assignments. The following array contains all the Place objects input from a file:

```java
ArrayList<Place> places = new ArrayList<Place>();
```

(A) Write the complete definition of a function called searchAll(...) that, given a town and a state, searches through the places list as defined above (and in your assignments) to return a list of all the zip codes used for that town and state. For example, for

```
town = “Cambridge”;
state = “MA”;
```

it will return a list containing the following zip codes (all of which belong to Cambridge, MA):

02138
02139
12140
02141
02142

**Note:** Only the function specified needs to be defined.
Given the class `Place` as defined in your assignments. The following array contains all the `Place` objects input from a file:

```java
ArrayList<Place> places = new ArrayList<Place>();
...
// place is now filled up with place objects.
```

(A) Write the complete definition of a function called `searchAll(...)` that, given a town and a state, searches through the `places` list as defined above (and in your assignments) to return a list of all the zip codes used for that town and state. For example, for

```
town = "Cambridge";
state = "MA";
```

it will return a list containing the following zip codes (all of which belong to Cambridge, MA):

```
02138
02139
12140
02141
02142
```

Note: Only the function specified needs to be defined.

```java
public static ArrayList<String> searchAll(ArrayList<Place> p, String town, String zip) {
    ArrayList<String> result = new ArrayList<String>();
    for (Place pl : p) {
        if (town.equalsIgnoreCase(pl.getTown())
            && zip.equalsIgnoreCase(pl.getState())) {
            result.add(pl.getZip());
        }
    }
    return result;
}
```

Question 3 (10+5 = 15 points)

```java
public static ArrayList<String> searchAll(ArrayList<Place> p, String town, String zip) {
    ArrayList<String> result = new ArrayList<String>();
    for (Place pl : p) {
        if (town.equalsIgnoreCase(pl.getTown())
            && zip.equalsIgnoreCase(pl.getState())) {
            result.add(pl.getZip());
        }
    }
    return result;
}
```
Question 3 (10+5 = 15 points)

```java
public static ArrayList<String> searchAll(ArrayList<Place> p, String town, String zip) {
    ArrayList<String> result = new ArrayList<String>();
    for (Place pl : p) {
        if (town.equalsIgnoreCase(pl.getTown())
            && zip.equalsIgnoreCase(pl.getState())) {
            result.add(pl.getZip());
        }
    }
} // searchAll()
```

(B) For the above function, what will be its asymptotic time complexity?

\[ O(n), \text{ where } n = p.\text{size()} \]

Question 4 (5+5=10 points)

Consider the statements shown below:

```java
int N = ...;
ArrayList<Integer> L = new ArrayList<Integer>();
for (int i=1; i <= N; i++) {
    L.add(2*i);
}
```

A. Write commands below that use an iterator to print out all the items in the list to the console window (one item per line):

```java
Iterator<Integer> iter = new L.iterator();
while (iter.hasNext()) {
    System.out.println(iter.next());
}
```
Question 4 (5+5=10 points)

Consider the statements shown below:

```java
int N = ...;
ArrayList<Integer> L = new ArrayList<Integer>();
for (int i=1; i <= N; i++) {
    L.add(2*i);
}
```

B. Write commands below that use the enhanced for-loop to print out all the items in the list to the console window (one item per line):

```java
for (Integer x : L) {
    System.out.println(x);
}
```

Question 5 (10 points)

Implement the add() function, as defined by the List<E> interface, in a singly-linked list that has both a head and a tail in its definition. Assume any helper functions in MyList<E> class that were defined in your text/class are available, including functions for the Node<E> class. Draw before/after diagrams of the data structure:

```java
public class MyList<E> implements List<E> {
    private Node<E> head, tail;
    private int size;
    ...
```
Question 5 (10 points)

Implement the add() function, as defined by the List<E> interface, in a singly-linked list that has both a head and a tail in its definition. Assume any helper functions in MyList<E> class that were defined in your text/class are available, including functions for the Node<E> class. Draw before/after diagrams of the data structure:

public class MyList<E> implements List<E> {
    private Node<E> head, tail;
    private int size;
    ...

    public boolean add(E item) {
        ...
    } // add()
Question 5 (10 points)

Implement the add() function, as defined by the List<E> interface, in a singly-linked list that has both a head and a tail in its definition. Assume any helper functions in MyList<E> class that were defined in your text/class are available, including functions for the Node<E> class. Draw before/after diagrams of the data structure:

```java
public class MyList<E> implements List<E> {
    private Node<E> head, tail;
    private int size;
    ...

    public boolean add(E item) {
        ...
    } // add()
}
```

```java
public class MyList<E> implements List<E> {
    private Node<E> head, tail;
    private int size;
    ...

    public boolean add(E item) {
        if (size==0) {
            head = tail = newNode(item);
        } else {
            ...
        }
        size++;
    } // add()
}
```
Question 5 (10 points)

```java
public boolean add(E item) {
    if (size==0) {
        head = tail = newNode(item);
    }
    else {
        Node<E> n = newNode(item);
        tail.next = n;
        tail = n;
        size++;
    }
    return True;
}
```

Question 5 (10 points)

```java
public boolean add(E item) {
    if (size==0) {
        head = tail = newNode(item);
    }
    else {
        Node<E> n = newNode(item);
        tail.next = n;
        tail = n;
        size++;
    }
    return True;
}
```
Question 6 (10 points)

- Show the result of evaluating the postfix expressions:
  
  - $11 \ 9 - = ____2____$
  
  - $11 \ 9 - 4 * = ____8____$
  
  - $4 \ 3 + 2 - 6 \ 2 + * = ____40____$
  
  - $3 \ 4 \ 7 * 2 / + = ____17____$
  
  - $4 \ 2 \ 2 + * = ____16____$

Question 7 (5+2+3 = 10 points)

Consider the commands below:

```java
LinkedList<String> potusList = new LinkedList<String>();
for (president : presidents) {
    potusList.add(president);
}
```

A. What will be the result of:

- `potusList.size()` ___6___
- `potusList.empty()` ___False___
- `potusList.get(1)` 
  - "Bush41"
- `potusList.get(7)` IndexOutOfBoundsException
- `potusList.contains("Lincoln")` ___False___
Question 7 (5+2+3 = 10 points)

Consider the commands below:

```java
LinkedList<String> potusList = new LinkedList<String>();

String[] presidents = {"Reagan", "Bush41", "Clinton",
                       "Bush43", "Obama", "Trump"};

for (president : presidents) {
    potusList.add(president);
}
```

B. What will be output when the following code is executed:

```java
for (int i=0; i < potusList.size(); i++) {
    System.out.println(potusList.get(i));
}
```

Reagan
Bush41
Clinton
Bush43
Obama
Trump

C. Rewrite the commands above, using the enhanced for loop. Other than its appearance, is there a difference between these two ways of writing this loop?

```java
for (String p : potusList)
    System.out.println(p);
```

It is much more efficient (O(n) vs O(n²))
Question 8 (10 points)

The summation of the following series:

\[ \sum_{i=0}^{n} i = 0 + 1 + \cdots + n \]

can be recursively defined as:

\[ \text{sum}(n) = \begin{cases} 
0, & \text{if } n = 0 \\
n + \text{sum}(n-1), & \text{o/w} 
\end{cases} \]

Write a complete recursive Java function `sum(n)` that carries out this computation as described above.

```java
public static int sum(int n) {
    if (n == 0)
        return 0;
    else
        return n + sum(n-1);
} // sum()
```

Question 9 (5+5=10 points)

A. What is a static variable in a class? Describe one use for it.

Static variables are shared variables, aka class variables. They are different from instance variables since, there is only one “copy” of a static variable that all instance of the class share. They are used in any situation where all instances need to share the same value.

B. What is printed by the command below:

```java
int N = (int) (1 + Math.random(100));
System.out.println("I have " + N + (N == 1 ? "item" : "items") + ".");
```

Depends on the value of N.

- I have 1 item. When N=1
- I have X items When N=X, X>1